UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,190	11/29/2003	Douglas Wooff	50325-0842	9853

29989 7590 09/04/2007

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
09/04/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/725,190	WOOFF ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 July 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-56* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-56* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/12/07;8/6/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 7/13/07.

As indicated in Applicant's response, claims 1, 3, 5, 8-11, 17-33, 36, 38-43, 49-52 have been amended, claims 53-56 added. Claims 1-56 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mathur, USPN: 5,008,814 (hereinafter Mathur), in view of Vishwanath, USPN: 2005/0198629 hereinafter Vishwanath).

As per claim 1, Mathur discloses a method of software loading and initialization in a distributed network of nodes, the method comprising:

persistently storing, in a first storage of a master node, a plurality of software packages and a plurality of boot program (e.g. Fig. 3; N_S, N_o – col. 5, line 3-20; *ILP* – col 7, lines 40-64 – Note: non-volatile storage for keeping cutover software for ILP in case of need reads on storage of boot package for regressing – see col. 3 line 65 to col. 4, line 49), wherein the plurality software packages and boot programs will be used by the nodes in the distributed network;

persistently storing, in a second storage of the master node, software version and node type, information for each node in the distributed network (e.g. col. 3 line 65 to col. 4, line 49; *new version, identifies node ...privileges* - col. 5, line 3-30; col. 9, line 36 to col. 10, line 8; col.

3, lines 29-53 – Note: *topology* information plus version related package and node identification in a list being stored in a distinct location for cutback after a failure in a *softload* reads on a node type information persisted for supporting a trial version to be rolled back at a second storage);

receiving, at the master node, a request for a boot program and software packages from a node, in the distributed network, that is performing an initial boot based on the request (e.g. *message receiver* – Fig. 3; request – col. 9 lines 9-53);

the master node extracting the boot program and one or more software packages from the first storage (col. 5, line 3-30; col. 9, line 36 to col. 10, line 8; col. 7, lines 24-44 – Note: *preferred* software treated as software of some version that has been retained for some use);

delivering, to the node, the boot program and the one or more software packages (e.g. col. 5, line 31 to col. 6, line 54; step *202* - Fig. 2);

wherein the node stores the boot program (*initial boot program* – col. 3, line 32-41) and the one or more software packages in its local persistent storage;

wherein software version information is extracted from the one or more software packages and stored (e.g. col. 3 line 65 to col. 4, line 49; *version ...release number* - col. 9, lines 45-52 – Note: each node storing of a trial version at local NVRAM reads on reboots using ILP software being downloaded from the N_S master node – see Fig. 3-- with storing version thereof at the local node storage) in the local persistent storage; wherein the node reboots and executes the boot program (e.g. col. 7, lines 40-44) stored in the local persistent storage.

Mathur does not explicitly disclose persistently storing of software version information by the master node in a second storage; the master node determining software version

information of the node therefrom and retrieving, based on said software version information of the node, boot program or packages from first storage.

However, in order to provide the proper version of boot package to the requesting node, Mathur discloses server storing of information about the network node (e.g. col. 3, lines 21-32) purported for a update paradigm by which working versions from tried versions are selectively downloaded (see Summary; *version ...release number* - col. 9, lines 45-52), and in the course of which appropriate communication status and topological information are checked to ascertain that the target node environment is appropriately set for any boot program to be transmitted and operable therein (see Fig. 2; col. 5, line 3-30; col. 9, line 36 to col. 10, line 8); hence Mathur's emphasis on having proper information related to provisioning working versions of boot programs to a requesting node is strongly implied. Vishwanath, in a network based provisioning of boot programs analogous to Mathur's node boot program distribution paradigm, discloses server with persistent storage of version information to accommodate request from NW node (see Fileserver 307 – Fig. 3; Fig. 8, 10; para 0039-0040, pg. 3-4; TABLE 1, pg. 4) to support downloading of package or image for booting a target based on analysis of the target OS environment characteristics and extracting of device and/or version information (e.g. Fig. 6, 14; para 0117, pg. 11). It would have been obvious for one skill in the art at the time the invention was made to implement Mathur's master node storage of NW information in regard to provisioning of boot package version indicated via master task's tracking of proper version and release number as set forth above, so that a persistent storage of version information such as taught by Vishwanath can support Mathur's request fulfilling endeavor such as to retrieve proper node environment and version of package. One would be motivated to do so because this

(master node) persisted version information would enable efficient mapping of the specifics of the requesting node with previously stored package, typological, environmental or version information thereof, especially when this information is persisted for subsequent reuse to expedite the proper download as set forth above in Mathur's bidirectional communication (e.g. Fig. 2-5) thereby facilitate extraction of the proper boot programs for download and allowing the requesting node to dynamically obtain the most update version during a ongoing boot process.

Nor does Mathur explicitly disclose that the boot program is a boot image; nor does Mathur disclose based on the software version information of the node, extracting a boot image from the first storage and delivering such boot image from the master node to network node. But based on the message from a node to request a package requiring a ILP by Mathur, the need to extract boot program from a package being received for such request with which to effect an attempt to boot (see: *fails to IPL, trial use* -col. 7, lines 15-63) is strongly implied. Based on Vishwanath teaching of image for enabling client node to boot their system, it would have been obvious for one skill in the art at the time the invention was made to implement the boot programs as boot image because as purported by Mathur's method to provisioning node as per a request basis of selective versions of working components would be facilitated by just sending an image (as by Vishwanath); thereby would obviate extraneous storage of a complete set of operating system components intended just for *initial program load* (IPL) at the target node; and this is the very intent by Vishwanath to use mapping rules in conjunction with server's database of booting components (see Vishwanath: para 0005 to 0008, pg. 1).

As per claim 2, Mathur discloses wherein said node, based on a command from said master node, does not store the one or more software packages in the local persistent storage

device, allowing said master node to download test software packages (e.g. Fig. 2 – Note: master node transmitted version to local node reads on local node not having it stored locally prior to transmission – see step 204, step 211 – Fig. 2) to said node and temporarily run said node using the test software packages, and wherein when said node reboots, the test software packages will no longer exist on said node (step 210 – Fig. 2 – Note: trial run leading to a cutback reads on not having the bad trial software upon reboot – see col. 12, line 46-60 – where only one trial version remains after a failure and cutback).

As per claim 3, Mathur (in view of Vishwanath) discloses wherein retrieving software version information creates the preferred software version information from the second storage based on functional features specified in the request by said node (see query handler - Fig. 4; *new version, identifies node ...privileges* - col. 5, line 3-30; col. 9, line 36 to col. 10, line 8 – Note: retrieving of proper version of boot program reads on storing of version based on stored privilege and/or identification information of nodes).

As per claim 4, Mathur discloses wherein said node verifies the software version information with said master node (step 204, step 211 – Fig. 2).

As per claim 5, Mathur discloses wherein if said node has the correct software versions, then said node completes booting by executing one or more software packages stored in the local persistent storage (step 211, Fig. 2; col. 3 line 65 to col. 4, line 49).

As per claim 6, Mathur discloses if said node does not have the correct software versions, retrieving correct software packages from the first storage and the correct software packages to said node (e.g. check consistency -Fig. 2), wherein said node stores the correct

software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage.

As per claim 7, Mathur discloses wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration (e.g. hierarchical structure, privileges – col. 5, lines 3-27 – Note: structure representing grouping of nodes according to some particular privilege configuration reads on class of nodes of same configuration but different processor ; see col. 3, lines 29-53) and may have differing processor types.

As per claim 8, Mathur does not disclose wherein each of one or more software package contains version information, dependency information, and other metadata information pertaining to software in the package; however teaches about administrator action based on topology of common path to node for routing, and hierarchy of privileges of nodes and checksum of software destined for distribution; as well as version and release of downloaded component (e.g. col. 9, lines 45-52; col 3, line 15-32; col. 5, lines 3-27; consistency check-- Fig. 2); hence the dependency over the network topological path, the access privileges and new version software information needed would be suggestive of metadata being collected via an administrative or server task for provisioning the above knowledge to informatively support the node distribution. Based on the boot image and storage of target system version and environment information by Vishwanath (refer to claim 1) wherein assembling a boot image implicates analysis, extraction, validation and retrieval of dependency specifics needed for packing components for boot deployment within a particular node OS environment (see Vishwanath: Fig. 3-10), it would have been obvious for one skill in the art at the time the

invention was made to provide the administrator or server/source node– master node – with stored metadata relating to version information, dependency information as mentioned above so that this information be packed in a target image as taught by Vishwanath; because this metadata would support the consistency checking as endeavored by Mathur in view of the topology-based for optimizing routing resources, expediting of the desired version of upgrade, and also for identifying access privilege per nodes as mentioned above.

As per claim 9, Mathur (in view of Vishwanath) discloses wherein a boot image is customized for a particular type of node and provides basic low-level communications (*initial boot program* – col. 3, line 32-41; Fig. 2 – Note: checking of checksum by master node in view of software to boot the node reads on boot image having node identification and low-level instructions, because without node type specificity is provided no proper reboot would be possible)

As per claim 10, Mathur (in view of Vishwanath) discloses method of software loading and initialization in a distributed network of nodes, the method comprising:

persistently storing, in a first storage of a master node, software packages and boot programs, which software packages and boot programs will be used by the nodes in the distributed network;

persistently storing in a second storage of the master node, software version and node type, information for each node in the distributed network;

receiving, at the master node, a request for a boot images and software packages from a node, in the distributed network, that is based on the request;

retrieving and extracting a boot images and one or more software packages from the first storage;

delivering, to the node, the extracted boot images and one or more software packages;

all of which limitations having been addressed in claim 1.

Mathur does not explicitly disclose persistently storing of software version information by the master node in a second storage; the master node determining software version information of the node therefrom and retrieving, based on said software version information of the node, boot program or packages from first storage. Nor does Mathur explicitly disclose that the boot program is a boot image and based on said software version information, extracting a boot image from the first storage and delivering such boot image from the master node to network node.

But these limitations have been addressed in claim 1 above using Vishwanath and Mathur.

As per claim 11, Mathur (in view of Vishwanath) discloses wherein said node stores the extracted boot image and one or more software packages in its local persistent storage (Fig. 1; col. 3 line 65 to col. 4, line 49) and wherein software version information is extracted from the one or more software packages and stored in the local persistent storage (e.g. col. 7, lines 32-53;col. 6, lines 41-54 – Note: packet reception of new software and for ILP for storage at node **reads on** extraction of package received based on version, checksum, packet time and topology of nodes etc. ; see col. 3, lines 29-53)

As per claims 12-14, refer to claims 2, 4-5 respectively.

As per claims 15-16, refer to claims 6-7, respectively.

As per claims 17-18, refer to the rationale of claim 8 and claim 9, respectively.

As per claim 19, Mathur (in view of Vishwanath) discloses boot images, software packages, and node information; and placing the boot programs and software packages in the first storage and the node information in the second storage on said master node (refer to claim 10); but Mathur does not explicitly disclose executing a composite image that is installed by a user onto said master node to create boot programs and packages and information; nor does Mathur disclose creating a subset of boot images, subset of plurality of software packages and placing these subset in first and second storage. The creation of boot program and node information being packaged for being extracted is disclosed in Mathur (re claim 10); and the administrative action to maintain version information and package for NW nodes update using operator's distribution command is taught (see *topology information, maintained* - col. 3, lines 29-53; col. 4, lined 64 to col. 5, line 27). Based on the creation of an image in Vishwanath's wherein package contents to support node booting are created via executing a rule-based validation and target analysis to generate a image (see Vishwanath: Fig. 3-10), the limitation of user's installing at the master node and executing a composite program (or rule script) from an incremental obtaining of subsets of package requirement (Vishwanath – see para 0126 to para 0133, pg. 11-14) and image to create the final image as taught in Mathur would have been obvious; that is, one of ordinary skill in the art would be motivated to provide a administrating tool by Mathur and using Vishwanath's teachings to support necessary files extracted from a persistent storage of image subsets (Vishwanath: Fileserver 307 – Fig. 3; Fig. 8, 10; para 0039-0040, pg. 3-4; TABLE 1, pg. 4) to support a incremental gathering of a programmatic content inside each boot image (see Vishwanath: Fig. 4, 6, 12), provide execution by a composite

process (see Fig. 14-15) by which the necessary boot programs, packages and type information (e.g. persisted in Mathur's topology information) stored at the master node to create the targeted node-specific boot image as set forth in the rationale of claim 10 or claim 1.

As per claim 20, refer to claim 3.

As per claim 21, Mathur (in view of Vishwanath) discloses a computer-readable medium carrying one or more sequences of instructions for software loading and initialization in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps recited in claim 1; hence the rejection for such steps will incorporate the corresponding rejection as set forth therein respectively.

As per claims 22-27, refer to claims 11-16, respectively.

As per claim 28, refer to claim 17.

As per claims 29, 31, refer to claims 18, 20, respectively.

As per claim 30, refer to claim 19.

As per claim 32, Mathur (in view of Vishwanath) discloses an apparatus of software loading and initialization in a distributed network of nodes, comprising a master node and storage means with node information for supporting network node request and delivery including boot image and software packages as recited in claim 1; hence the rejection for all such limitations or means will incorporate the corresponding rejection as set forth therein respectively.

As per claims 33-38, refer to claims 22-26, respectively.

As per claim 39, refer to claim 28.

As per claims 40, 42, refer to claims 29, 31, respectively.

As per claim 41, refer to claim 30.

As per claim 43, Mathur discloses a system for software loading and initialization in a distributed network of nodes, a node in the distributed network;

a first storage on the master node, wherein the first storage stores boot programs and software packages that nodes in the distributed network will use;

a second storage on the master node, wherein the second storage stores software version and node type information for each node in the distributed network; one or more processors on the master node;

one or more sequences of instructions which, when executed by the one or more processors, cause the one or more processors to perform:

receiving a request for a boot program and software packages from the node that is performing an initial boot;

based on the request, retrieving preferred software version for the node from the second storage; extracting a boot program and one or more software packages from the first storage; and delivering, to the node, the extracted boot program and one or more software packages;

one or more other processors on the node (e.g. Fig. 1);

one or more other sequences of instructions which, when executed by the one or more other processors, cause the one or more other processors to perform:

storing the extracted boot program and one or more software packages in local persistent storage of the node;

extracting software version information from the software packages; storing the software version information in the local persistent storage; executing the boot program, that is stored in the local persistent storage, to reboot the node;

all of which limitations being addressed in claim 1.

Mathur does not explicitly disclose persistently storing of software version information by the master node in a second storage; the master node determining software version information of the node therefrom and retrieving, based on said software version information of the node, boot program or packages from first storage. Nor does Mathur explicitly disclose that the boot program is a boot image and based on said software version information, extracting a boot image from the first storage and delivering such boot image from the master node to network node.

But these limitations have been addressed in claim 1 above using Vishwanath and Mathur.

As per claims 44-46, refer to claims 2, 4-5, respectively.

As per claim 47, Mathur discloses wherein the one or more sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to:

perform retrieving correct software packages from the first storage and sending the correct software packages to the node if the node does not have the correct software versions (re claim 6 or 15); and the one or more other sequences of instructions which, when executed by the one or more other processors, further cause the one or more other processors to

perform storing the correct software packages in the local persistent storage and executing the correct software packages stored in the local persistent storage to complete booting (re claim 6 or 15).

As per claims 48-50, refer to claims 7-9, respectively.

As per claim 51-52, refer to claims 19-20, respectively.

As per claim 53, Mathur (in view of Vishwanath) discloses by virtue of the obviousness rationale in claim 1, master node using the node type information of the node to determine the software version information of the node; that is, discovery capability of the provisioning server by Vishwanath – see para00 37-0040; Fig 6; para 0126 to para 0133, pg. 11-14 – to complement the version software for upgrade in node topology and version checking by Mathur as set in claim 1, would have motivated one of ordinary skill in the art to implement Mathur's server/master storage of NW topology so that this storage contains node information based on which to derive the proper and required boot components (e.g. proper version information) as discovered in Vishwanath's server storage system (see Vishwanath: Fig. 4).

As per claims 54-56, refer to the obviousness rationale of claim 53.

Response to Arguments

4. Applicant's arguments filed 2/13/07 have been fully considered but they are most the most part moot in view of the new grounds of rejection. As for some arguments related to Mathur deemed particularly relevant to the parts of the claims that remain unchanged, following are the Examiner's observation in regard thereto.

35 USC § 103 Rejection:

(A) Applicants have submitted that Mathur does not teach request for a boot image and software packages (Appl. Rmrks pg. 25). The argument is a repeat of an issue that had been raised in the previous response and in regard to which the Office Action had replied in the previous Examiner's Response to Arguments (refer to section C of said Response as per the Final Office Action mailed 4/25/07). Notwithstanding the lack of specifics in the claim language

to convincingly distinguish a recited 'request' from Mathur's message from a node for a version of boot program, the grounds of rejection has been clear on the boot image limitation. Indeed, this 'boot image' has been viewed as an obvious limitation thus requiring a combination of teachings; and accordingly, Applicants fail to acknowledge the obviousness rationale addressing the merits of this limitation. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

(B) The rest of the arguments revolve around the newly added limitations. New grounds of rejection will render these arguments moot. In all, the claims would stand rejected as set forth in the Office Action.

Interview Summary

5. The Applicant's representative, Daniel Ledesma, was approached on August 16, 2007 in order to reach an agreed upon language that would import certain features within some dependent claims into the context of the latest version of the independent claims; so that the changes could hopefully bring out some non-obvious aspect of the Invention over the prior art. But there was no agreement reached.

Conclusion

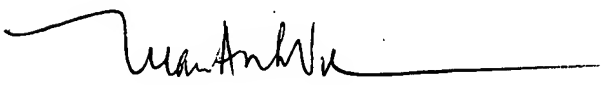
6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Tuan A Vu
Patent Examiner,
Art Unit 2193
August 29, 2007